

# Aula 11

## Dispositivos Lógicos Programáveis

**SEL 0414 - Sistemas Digitais**

**Prof. Dr. Marcelo Andrade da Costa Vieira**

# 1. Características Gerais

- PLD – *Programmable Logical Device*;
- Agrupa um grande número de portas lógicas, flip-flops e registradores conectados em um único dispositivo;
- A função específica do circuito é determinada definindo-se as conexões internas que devem ser abertas ou fechadas.

## 2. Implementação de Circuitos Digitais

### CIs convencionais

- ➔ necessitam de um processo de fabricação especial que requer máscaras específicas para cada projeto
- ➔ tempo de desenvolvimento é longo e os custos são altos
- ➔ utilizados em aplicação de grande volume de produção.

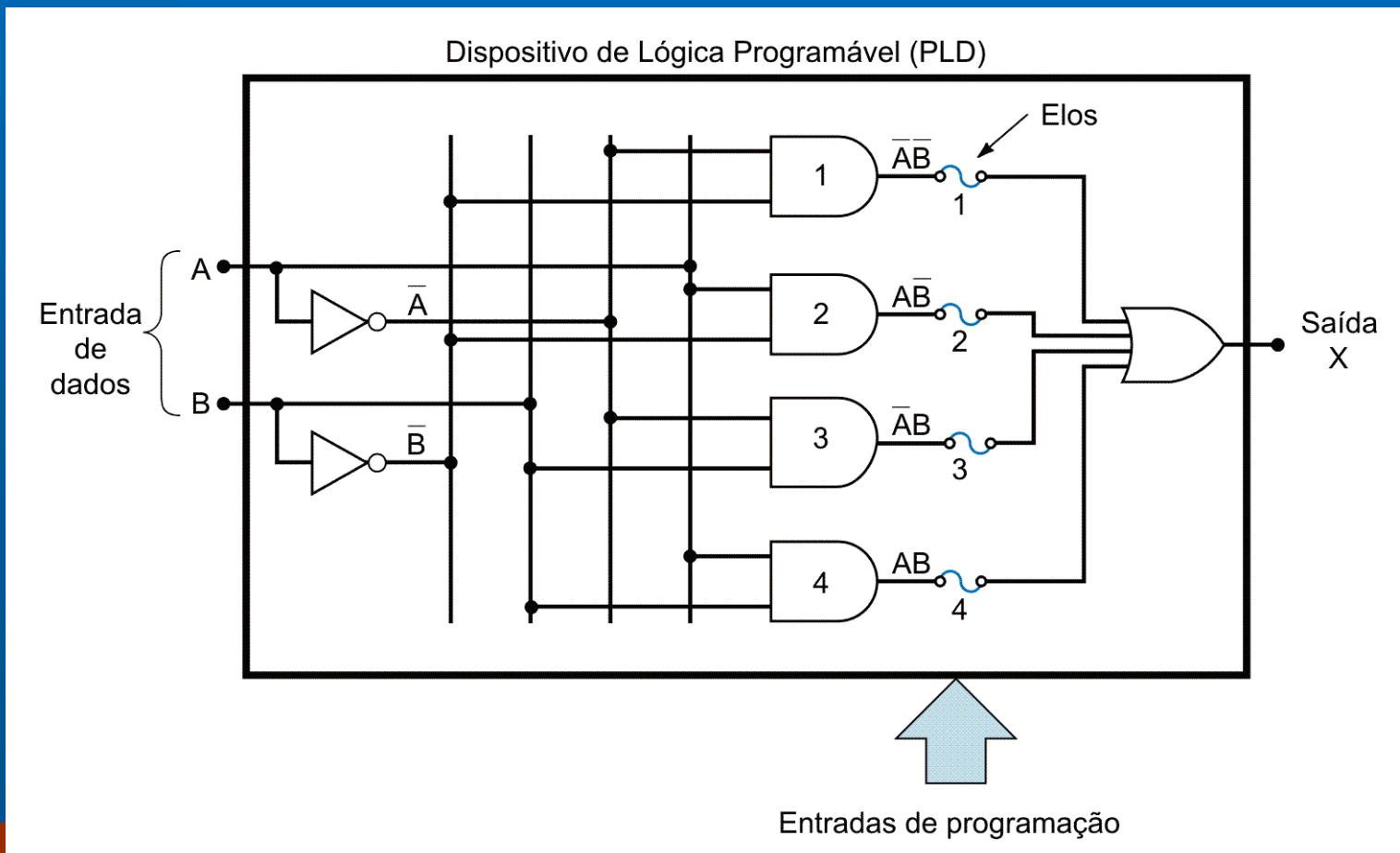
## 2. Implementação de Circuitos Digitais

### Cl's PLD

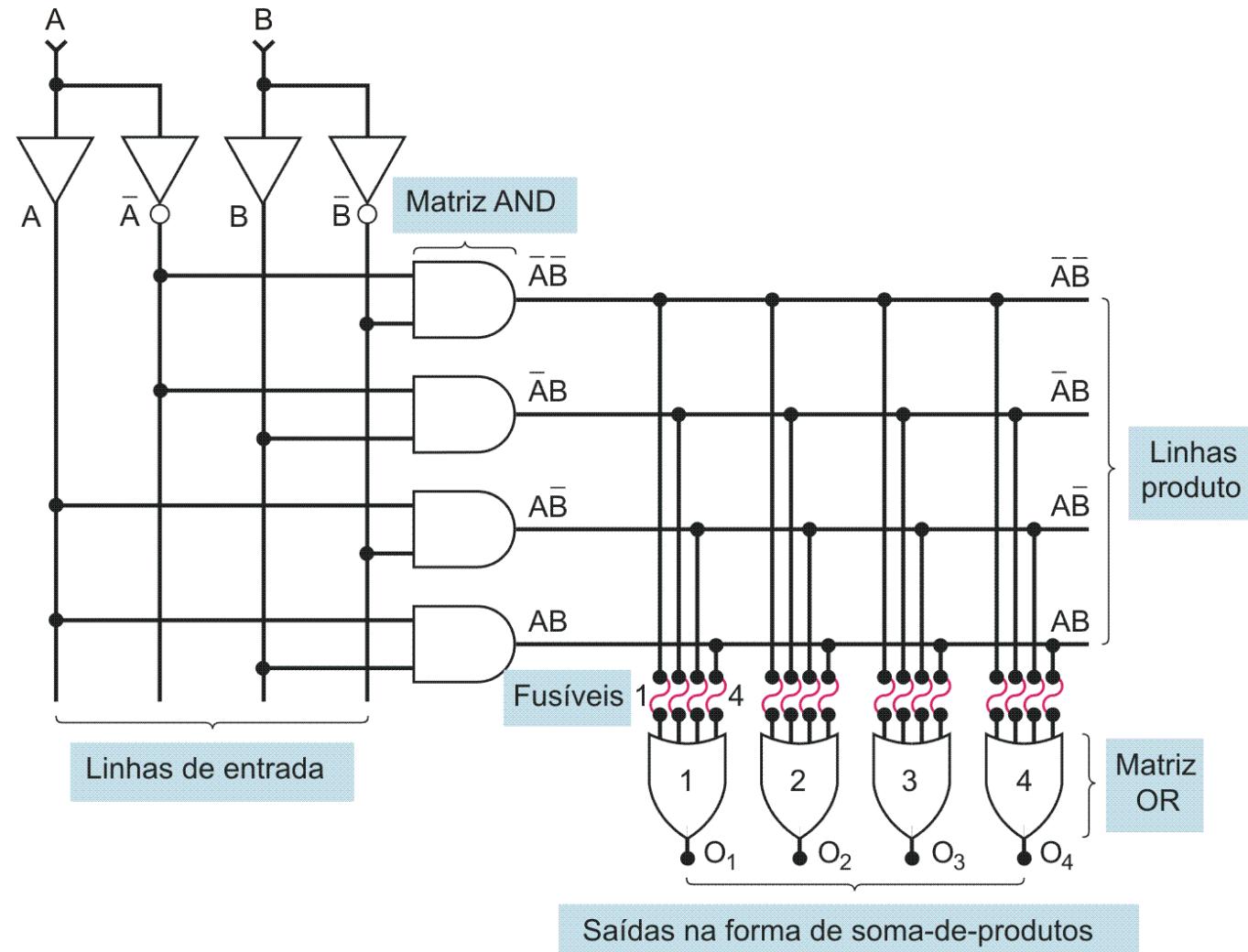
- ➔ Eliminação do processo de fabricação especial
- ➔ Processo de fabricação mais rápido e barato
- ➔ Programado pelo usuário
- ➔ Baixo custo e tempo curto de projeto
- ➔ Menor espaço ocupado nas placas
- ➔ Alteração na interconexão do dispositivo  
(programação) via *linguagem de descrição de hardware (HDL)*

## Arquitetura Básica:

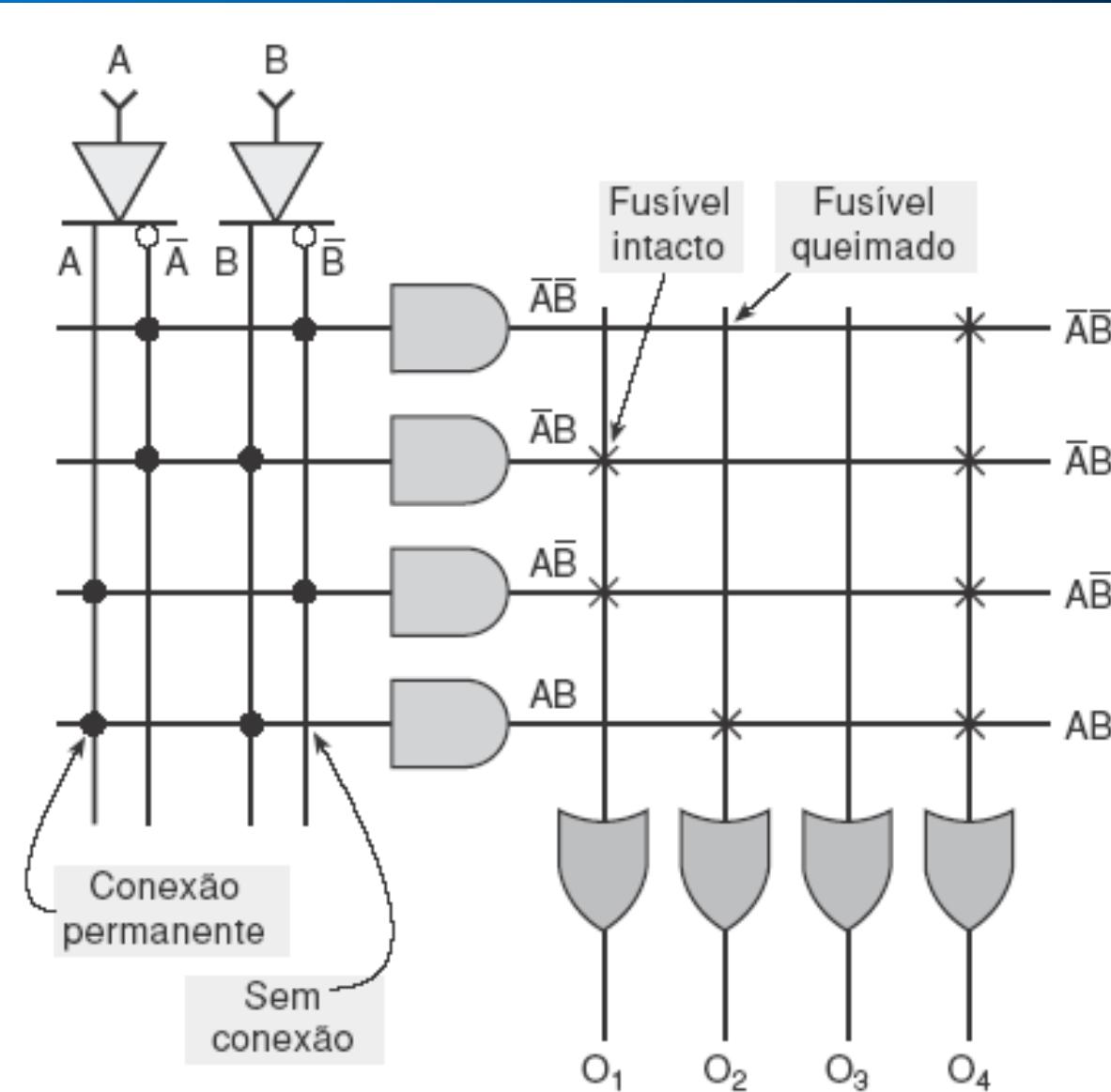
- Colocação de muitas portas lógicas num único CI
- Controlar eletronicamente a conexão entre elas.



# Simbologia Básica



# Simbologia Simplificada



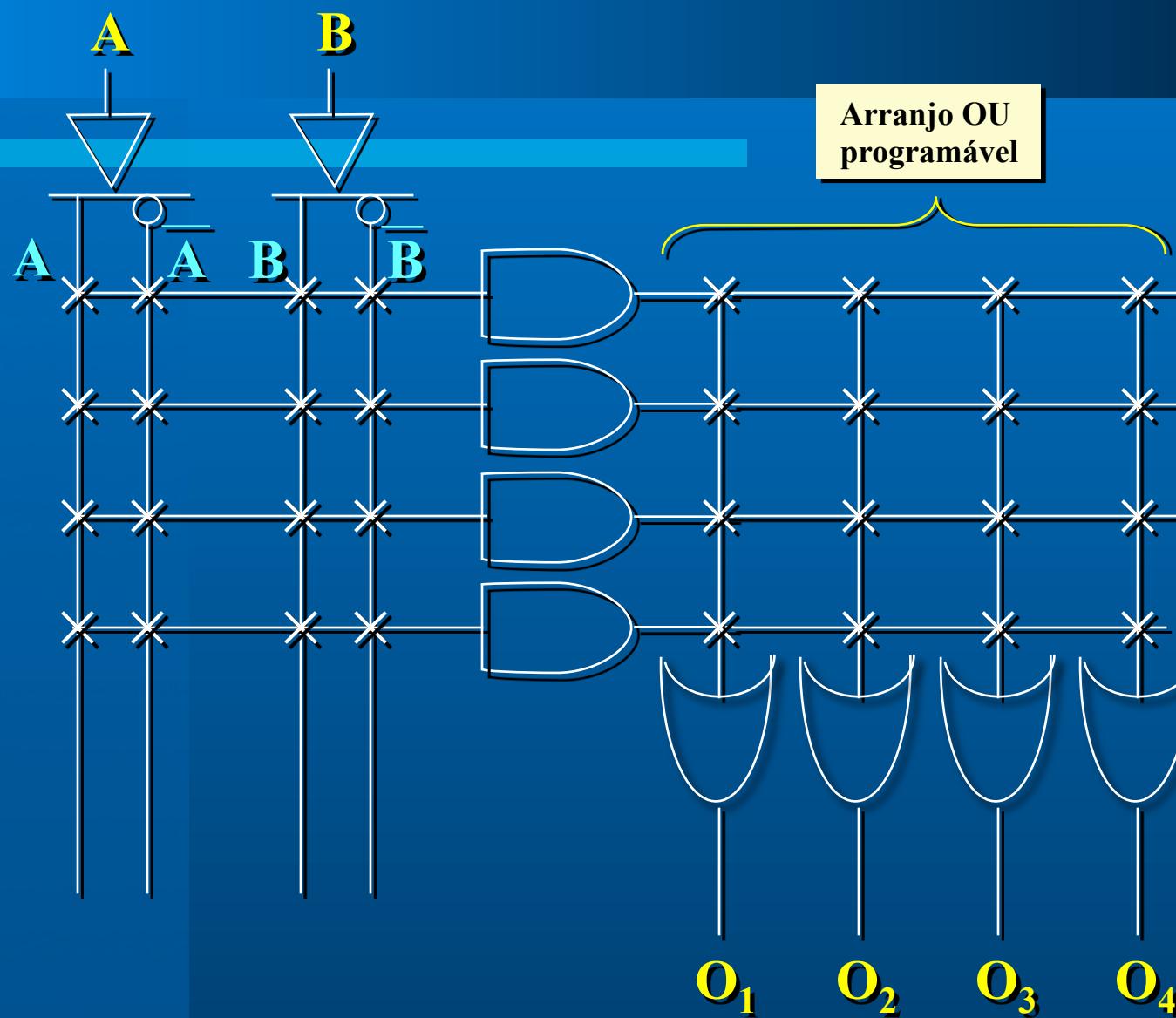
## 4. Arquiteturas de PLDs

### FPLAs ( Field Programmable Logic of Arrays):

- 1º. Dispositivo desenvolvido para a implementação de circuitos lógicos (década de 70)
- Dois níveis de portas lógicas programáveis, um de portas E e outro de OU
- Apresentam alto custo e desempenho ruim em termos de velocidade.
- Pouco utilizado

# Representação Típica

FPLA

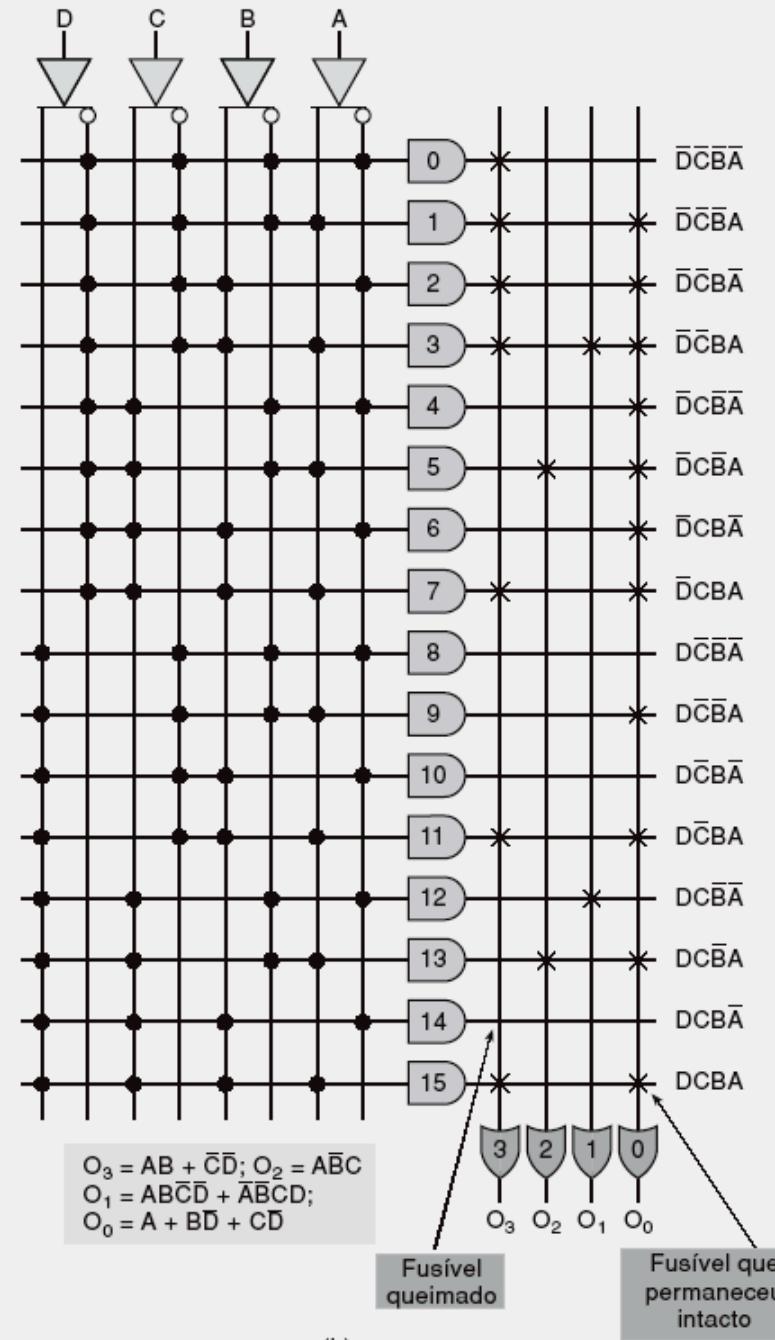
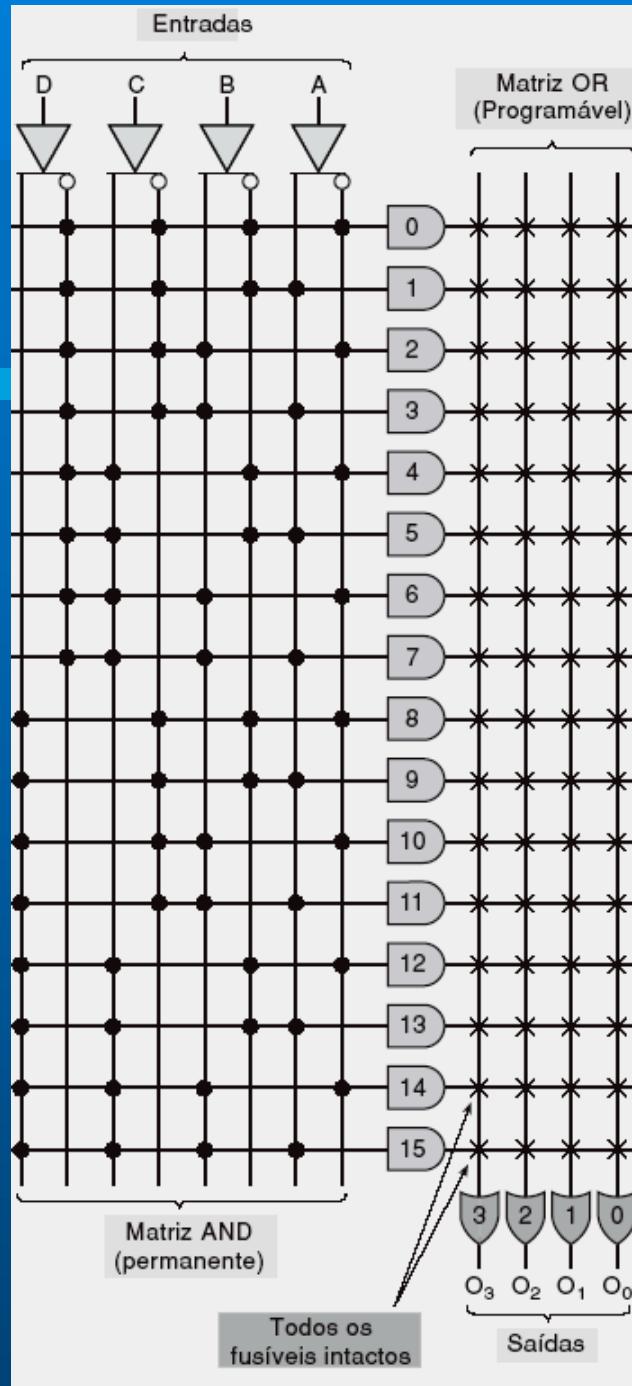


## 4. Arquiteturas de PLDs

### PROM ( Programmable Read Only Memory):

- Memórias PROM podem ser utilizadas como PLDs, já que sua arquitetura é semelhante
- Conexão permanente de portas AND e conexões programáveis nas portas OR
- Dificuldades no projeto de equações booleanas simples
- Só pode ser programado uma vez

# PROM



## 4. Arquiteturas de PLDs

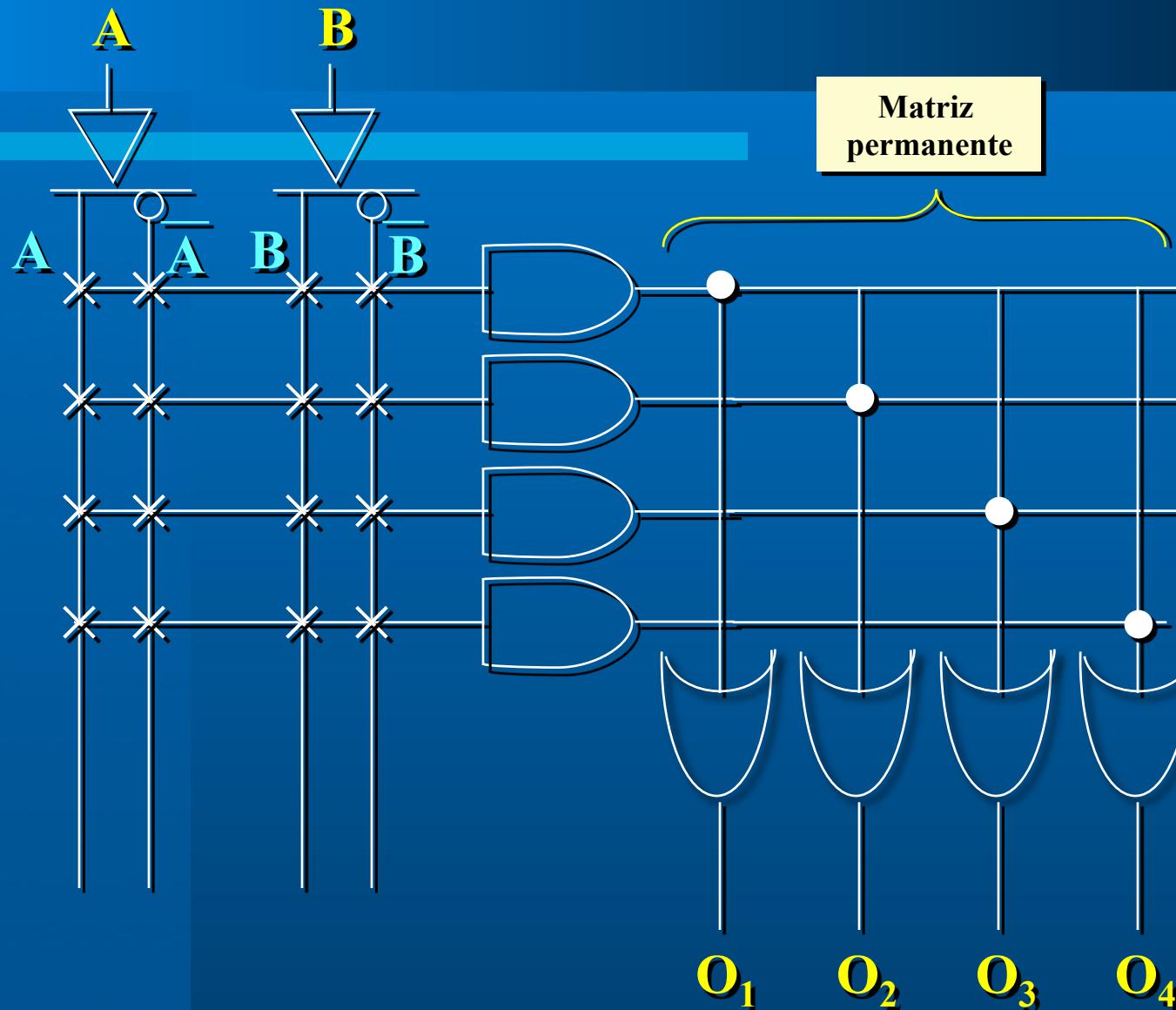
### PAL (Programmable Array Logic):

- Desenvolvidos para superar as deficiências das FPLAs e das PROMs
- Um único nível de programação – portas E programáveis alimentando portas OU permanentes
- São produzidas com diferentes quantidades de entradas e saídas
- Só pode ser programado uma vez

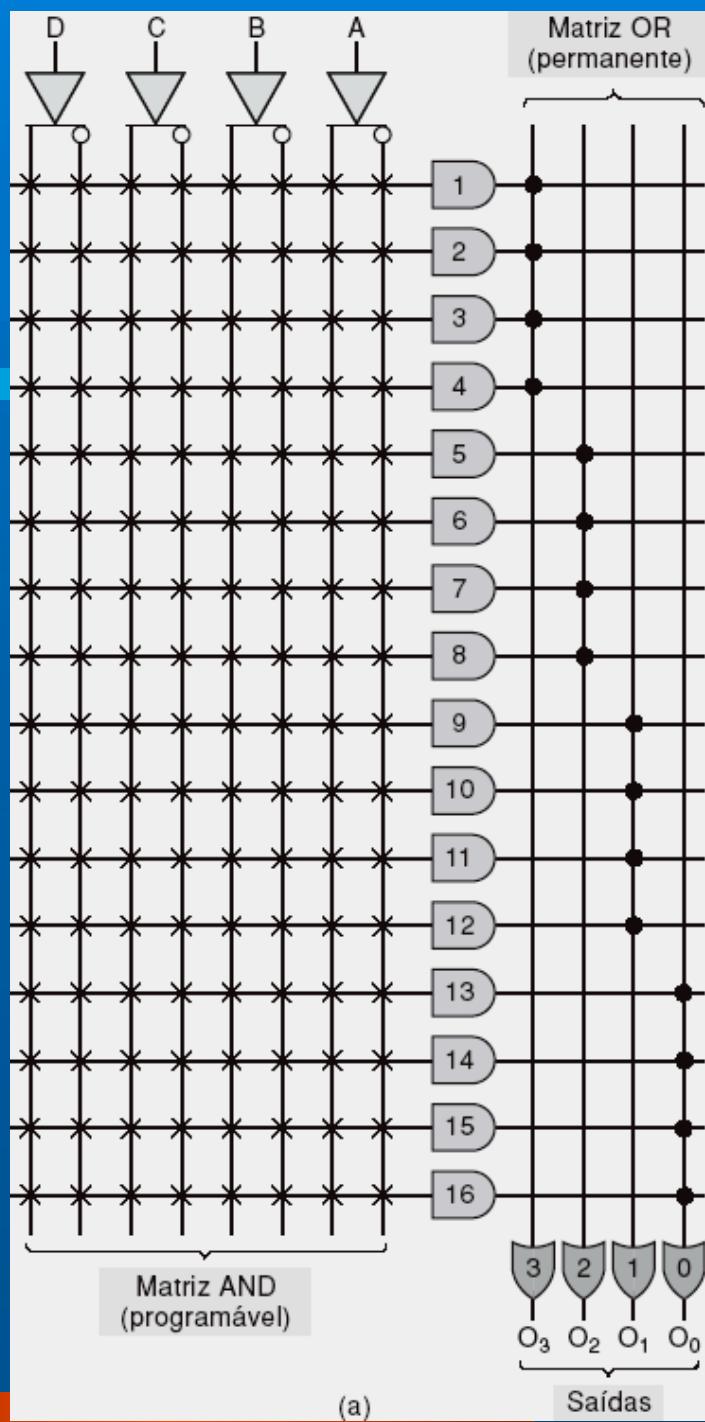
**OBS.:** geralmente apresentam flip-flops conectados às saídas das portas OU para que circuitos sequenciais possam ser implementados.

# Representação Típica

PAL

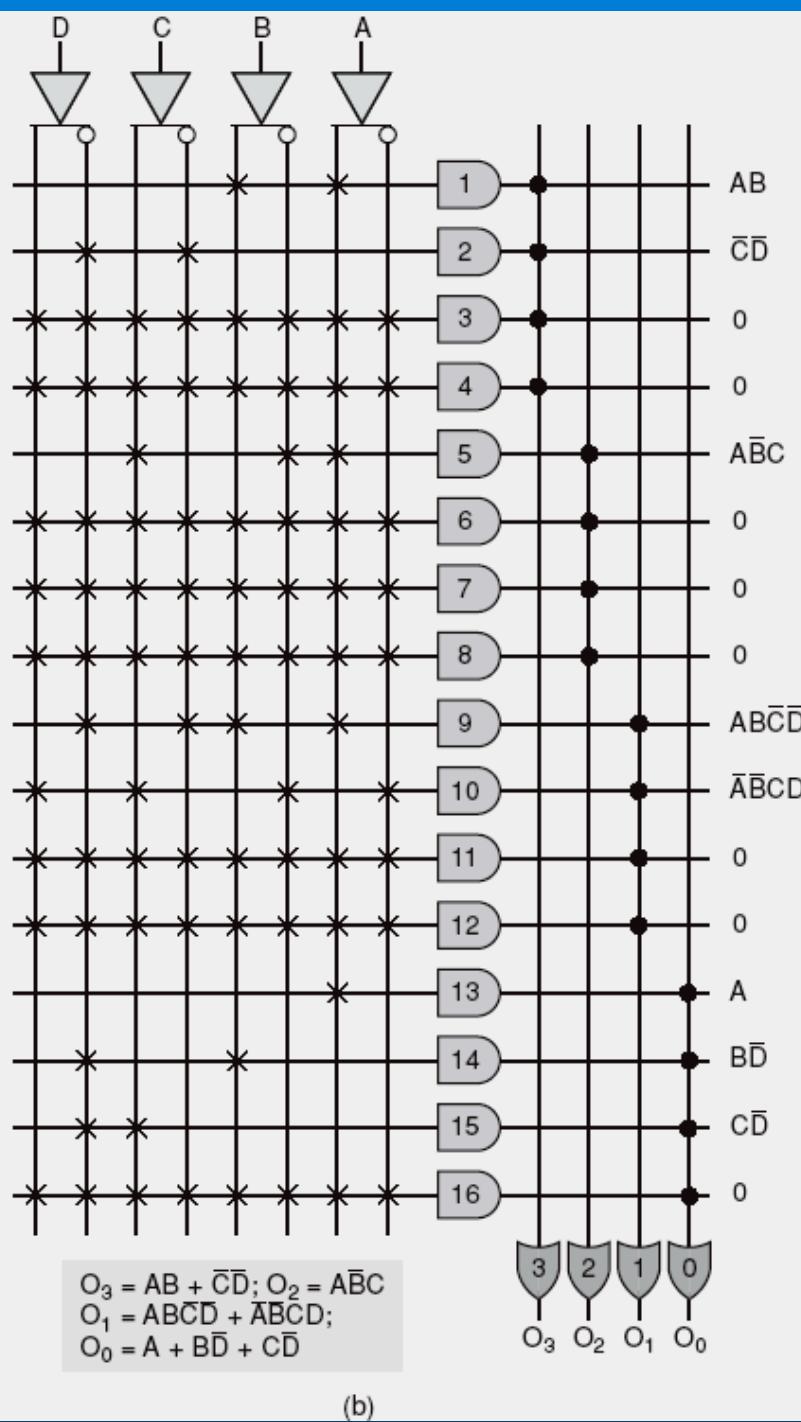


# PAL



(a)

Saídas



(b)

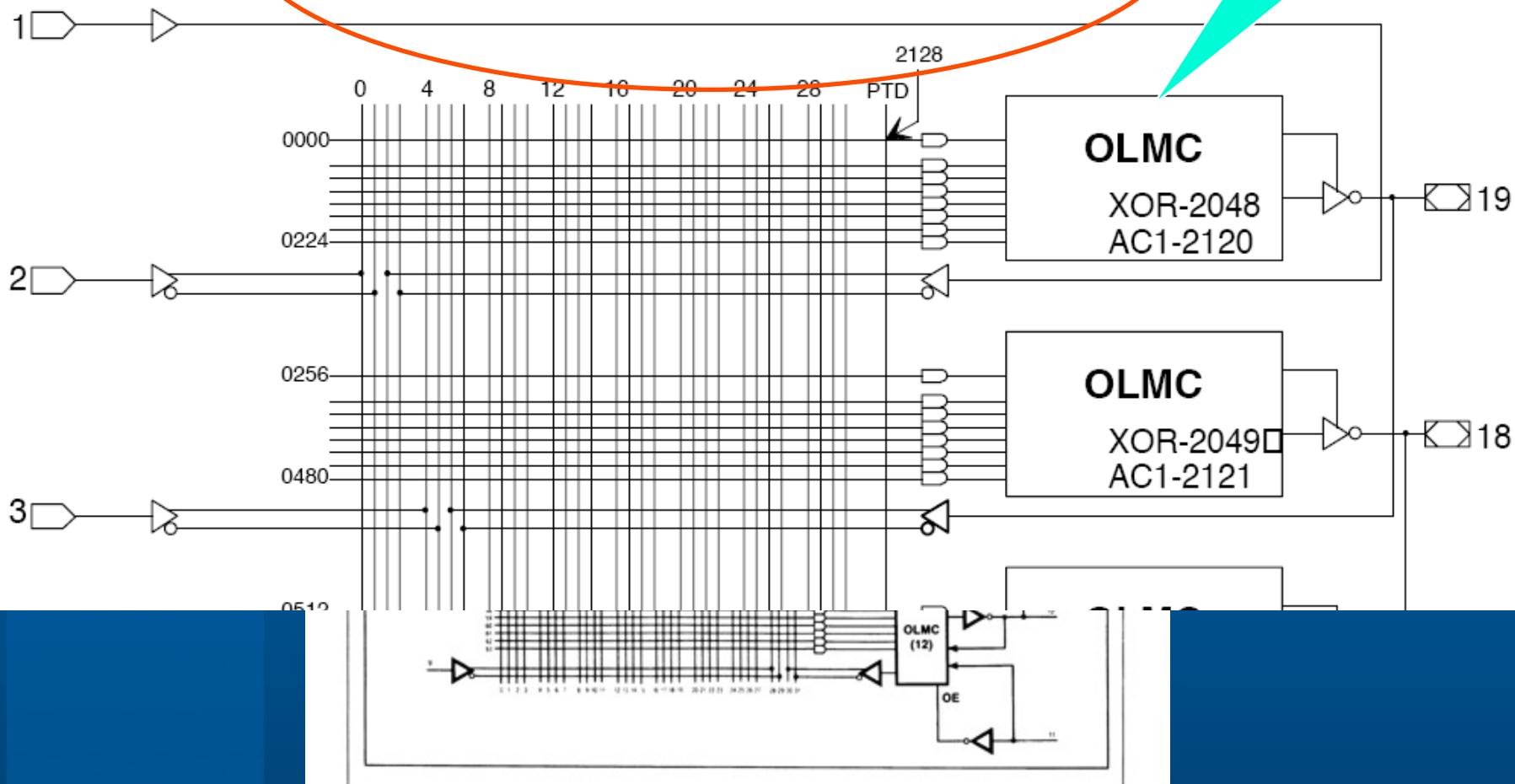
## 4. Arquiteturas de PLDs

### GAL (Generic Array Logic):

- Arranjo lógico genérico
- Arquitetura semelhante a PAL com compatibilidade de pinos
- Ao invés de fusíveis, possui memória EEPROM
- Podem ser programadas e apagadas diversas vezes (+ de 100)

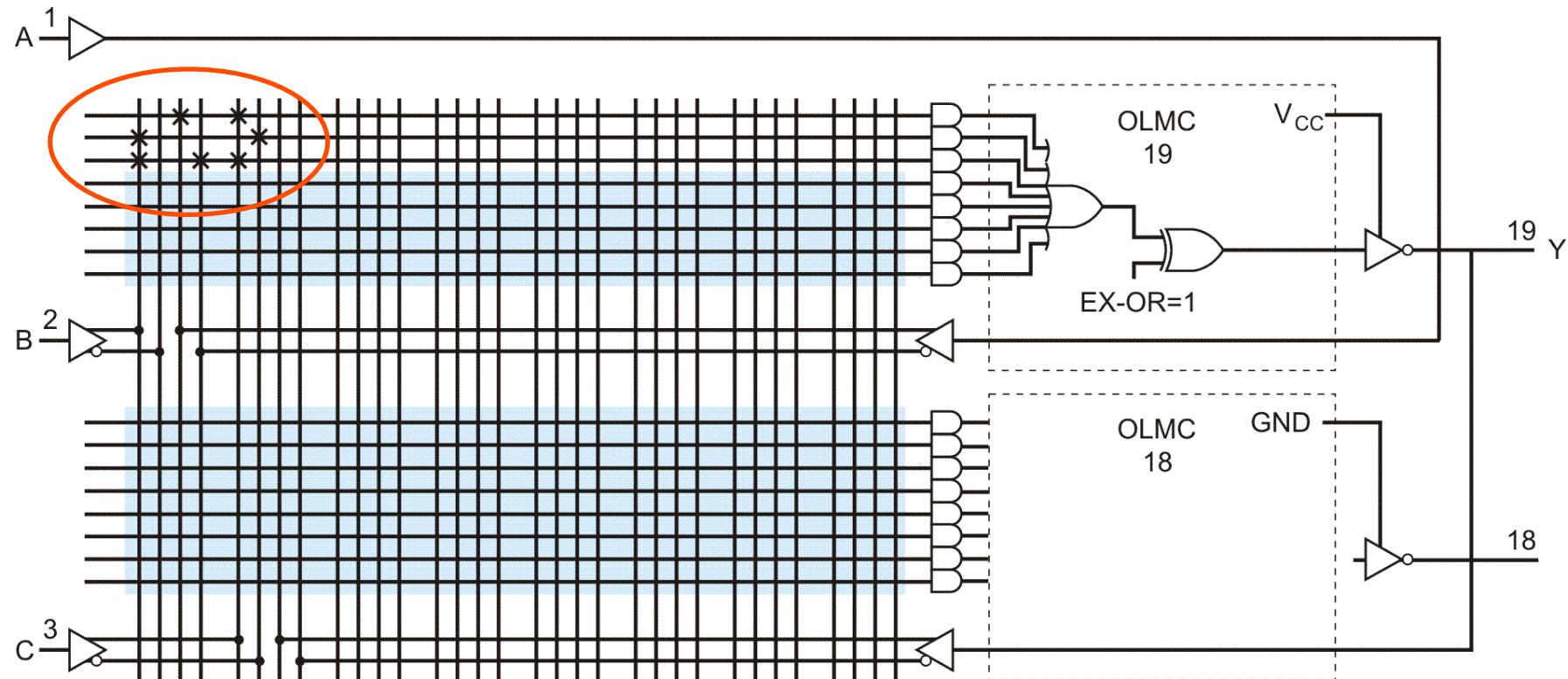
Macrocélula  
de saída  
(programável)

GAL16V8 LOGIC DIAGRAM



## Exemplo:

→ Implementação de uma lógica combinacional simples



$$Y = AC + \bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$$

## 6. Outros grupos de PLDs

### SPLDs (Simple Programmable Logic Devices):

- Categoria de todos os pequenos PLDs como PLAs, PALs
- Características mais importantes: baixo custo e alto desempenho.

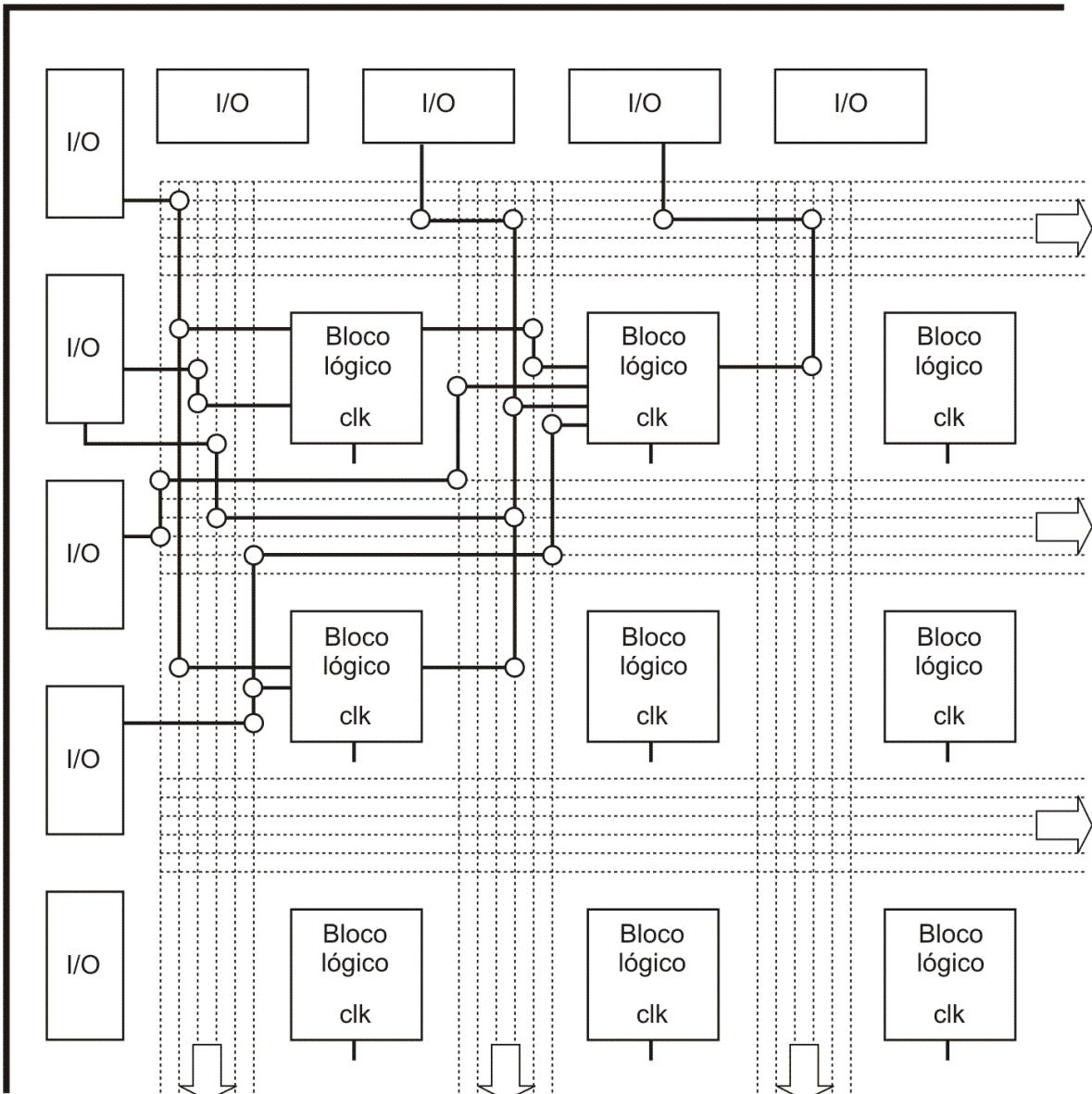
### CPLDs (Complex Programmable Logic Devices):

- Constituídos de múltiplos SPLDs integrados em um único chip
- Apresentam interconexões programáveis para conectar os blocos SPLDs
- Capacidade lógica de até 50 SPLDs típicos.

### FPGA (Field-Programmable Gate Array):

- Constituídos de um arranjo de elementos de circuitos não conectados – os blocos lógicos – e recursos de interconexão
- Configuração: programada pelo usuário.

# FPGA



○ Interconexão programável  
— Segmento de conexão  
- - - Caminho para interconexão

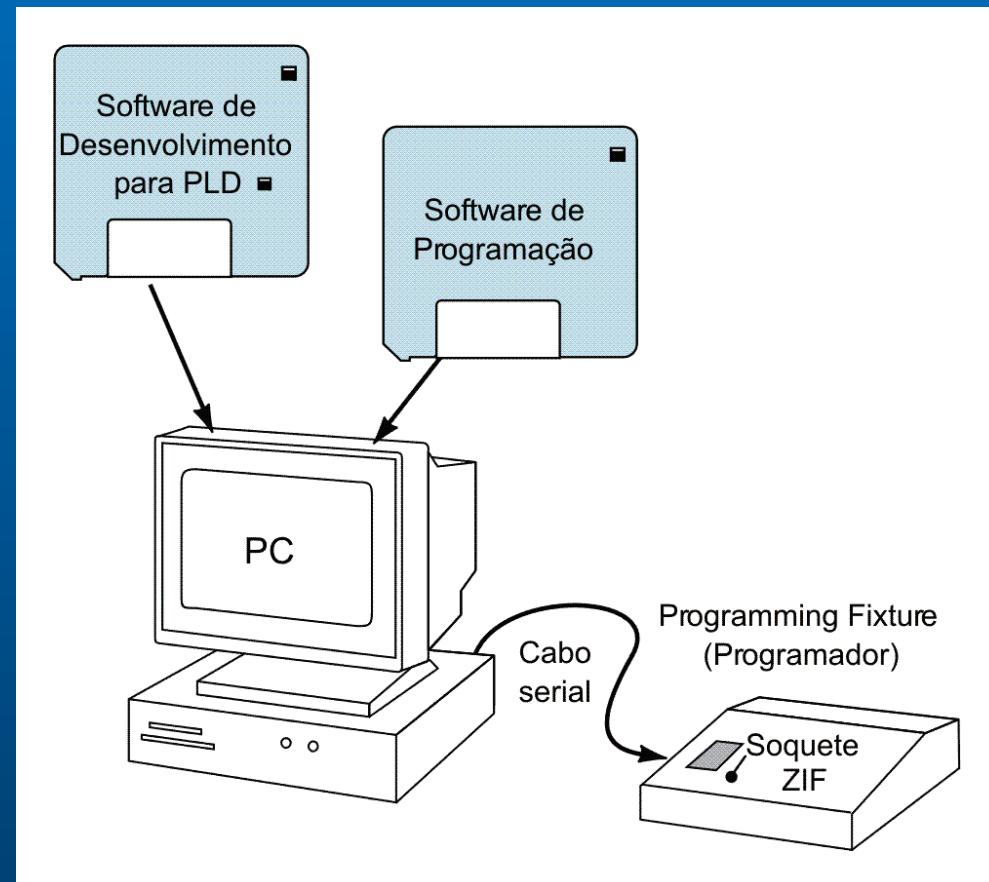
NOTA: As entradas de clock  
podem ter caminhos  
especiais de interconexão  
com baixa inclinação.

# PLD

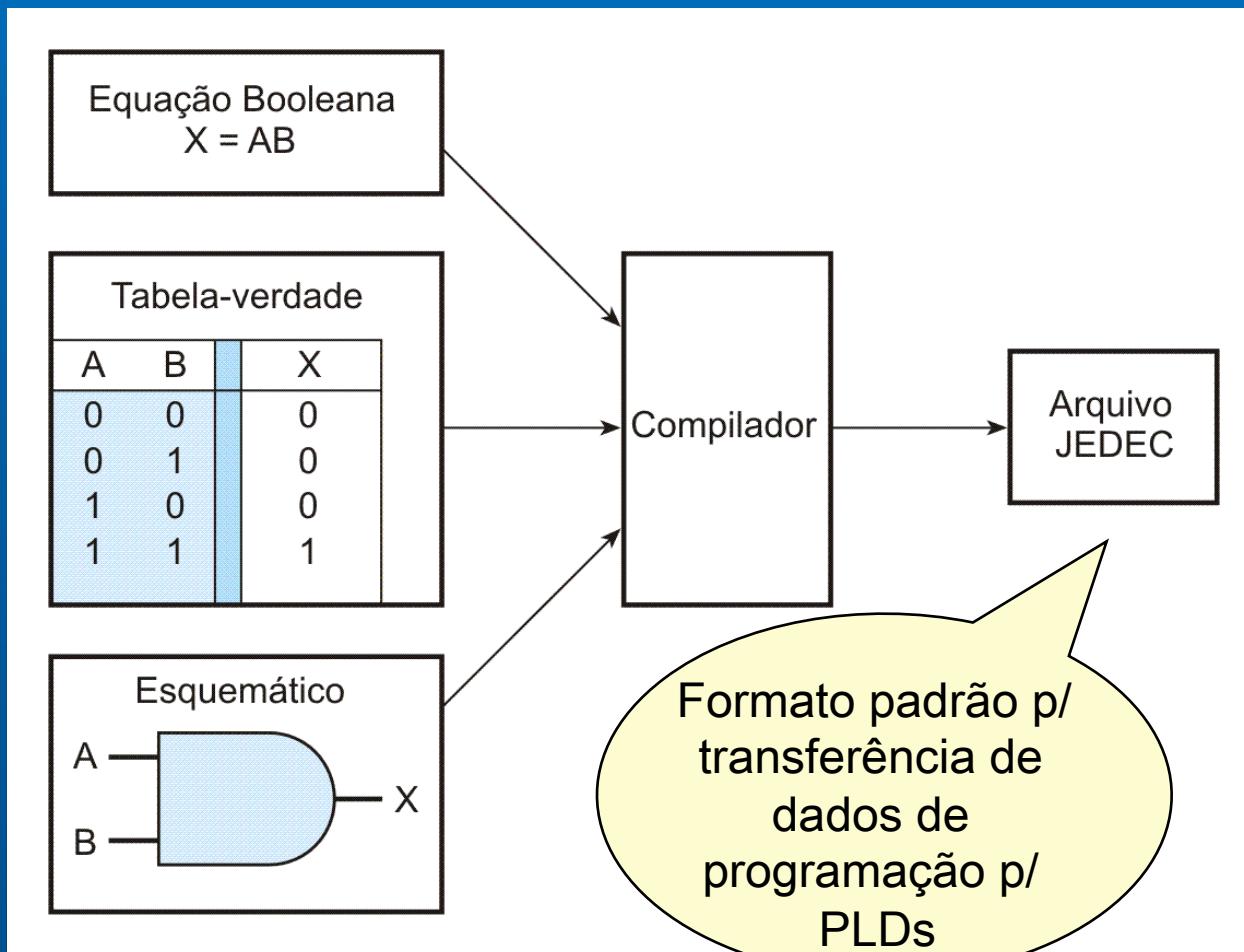
## Modo de programação do chip PLD:

Dispositivo Programador ➔ conectado ao PC (software com as bibliotecas dos PLDs disponíveis) ➔ software gera o MAPA DE FUSÍVEIS

Software permite configurar o Programador com os dados do PLD e checar as conexões



# Programação



## Programação:

- ➔ Linguagens de Descrição de Hardware - HDL (*Hardware Description Language*)
- ➔ VHDL – Para dispositivos VHSIC (Very High Speed Integrated Circuit)
- ➔ Vários compiladores (HDL → Net List):
  - ➔ ABEL (Data I/O Corporation)
  - ➔ CUPL (Logical Devices Inc.)

# CUPL (*Universal Compiler for Programmable Logic*)

Função	Operador	Formato CUPL	Formato convencional
E	&	A & B	A . B
OU	#	A # B	A + B
NÃO	!	!A	$\overline{A}$
OU-EXCL	\$	A \$ B	A⊕B

$$S = \overline{\overline{A}}BC + A\overline{\overline{B}}C + AB\overline{\overline{C}}$$

$$S = !A&B&C#A&!B&C#A&B&!C$$



# CUPL (*Universal Compiler for Programmable Logic*)

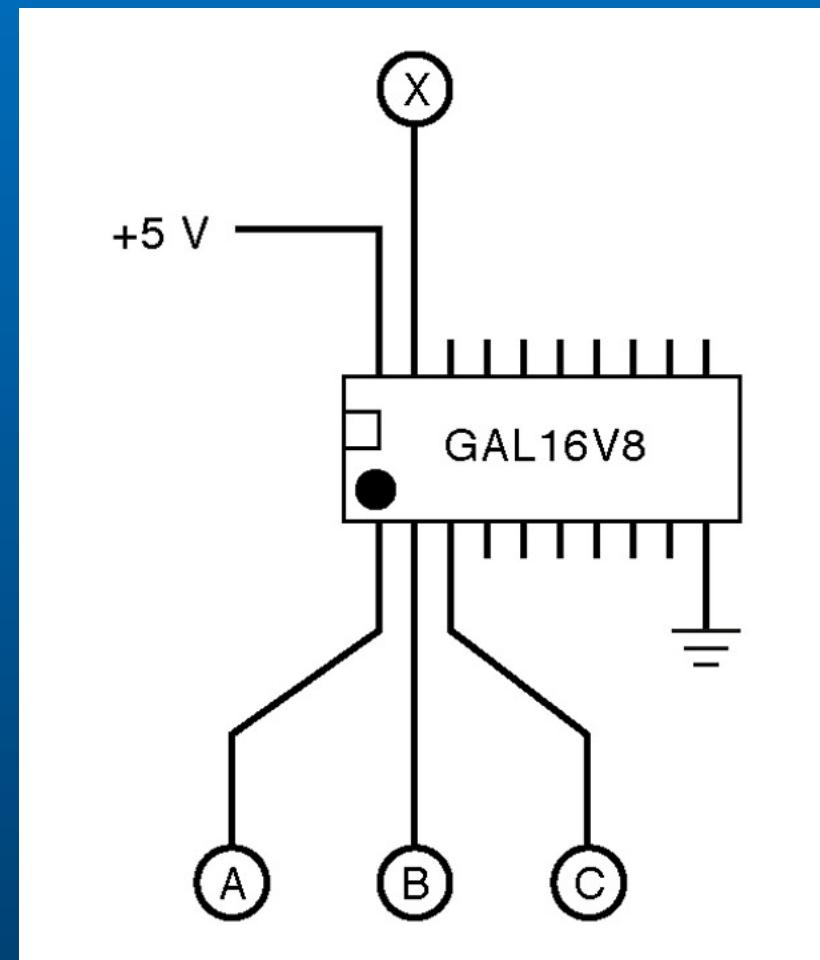
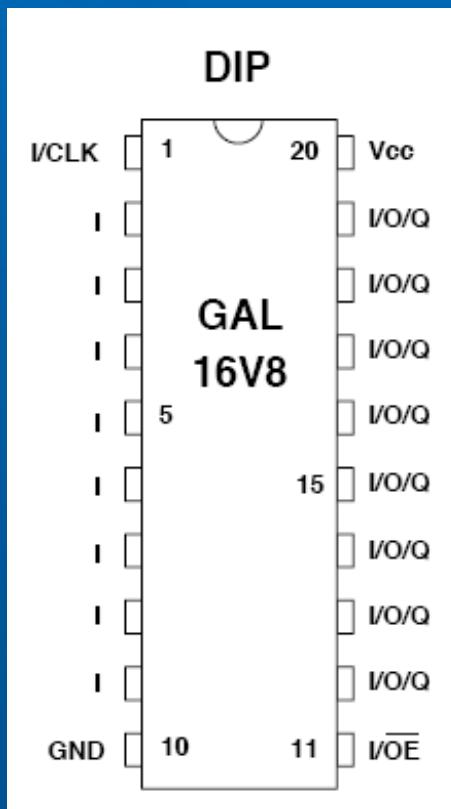
- Cabeçalho
- Especificação da entrada (Por ex.: pino 1: A; pino 2: B...)
- Especificação da saída (Por ex.: pino 19: S)
- Especificação da descrição do *Hardware*  
(Por ex.: implementação da equação booleana):

$$S = !A \& B \& C \# A \& !B \& C \# A \& B \& !C$$

# GAL

**Exemplo:**

- ➔ GAL16V8 – *Lattice Semiconductor Corporation*
- ➔ Possui 8 entradas e 8 saídas básicas configuráveis



CUPL defines these header categories. Omitting them produces warning messages. Misspelling them produces error messages.

The designer chooses these entries for documentation purposes.

Name	combo.pld	:
Partno	1234567	:
Date	June 2	:
Revision	02	:
Designer	N.S.Widmer	:
Company	Purdue University;	
Assembly	Chapter 4	:
Location	Chapter 4	:
Device	G16V8	:
Format	j	:

Semicolons are required at the end of each line.

All text between /\* and \*/ serves as comments. You can type anything here.

/\*JEDEC

\*/

/\* Simple Combinational Logic Example \*/

/\* Inputs Specification \*/

pin 1 = A ;  
Pin 2 = B ;  
pin 3 = C ;

The designer chooses input and output signal names.

/\* Outputs Specification \*/

pin 19 = X ;

Output variable on left-hand side

/\* Hardware Description Section \*/

X = !A&B&C#A&!B&C#A&B&!C#A&B ;

Sum of Products expression for X



OR  
AND  
NOT

## 7. Modo de entrada por conjuntos (*field*)

$$X = A \cdot B$$

$$A = [A_3 \ A_2 \ A_1 \ A_0]$$

$$B = [B_3 \ B_2 \ B_1 \ B_0]$$

$$X = [X_3 \ X_2 \ X_1 \ X_0]$$



$$\text{field } A = [A3..0]$$

$$\text{field } B = [B3..0]$$

$$\text{field } X = [X3..0]$$

$$X = A \& B$$

## 8. Modo de entrada por tabela verdade

Table [A,B] => X

```
{ [0,0] => 0;
  [0,1] => 1;
  [1,0] => 1;
  [1,1] => 0; }
```

(a) Set format

Table [A,B] => X

```
{ 'b'00 => 0;
  'b'01 => 1;
  'b'10 => 1;
  'b'11 => 0; }
```

(b) Binary set values

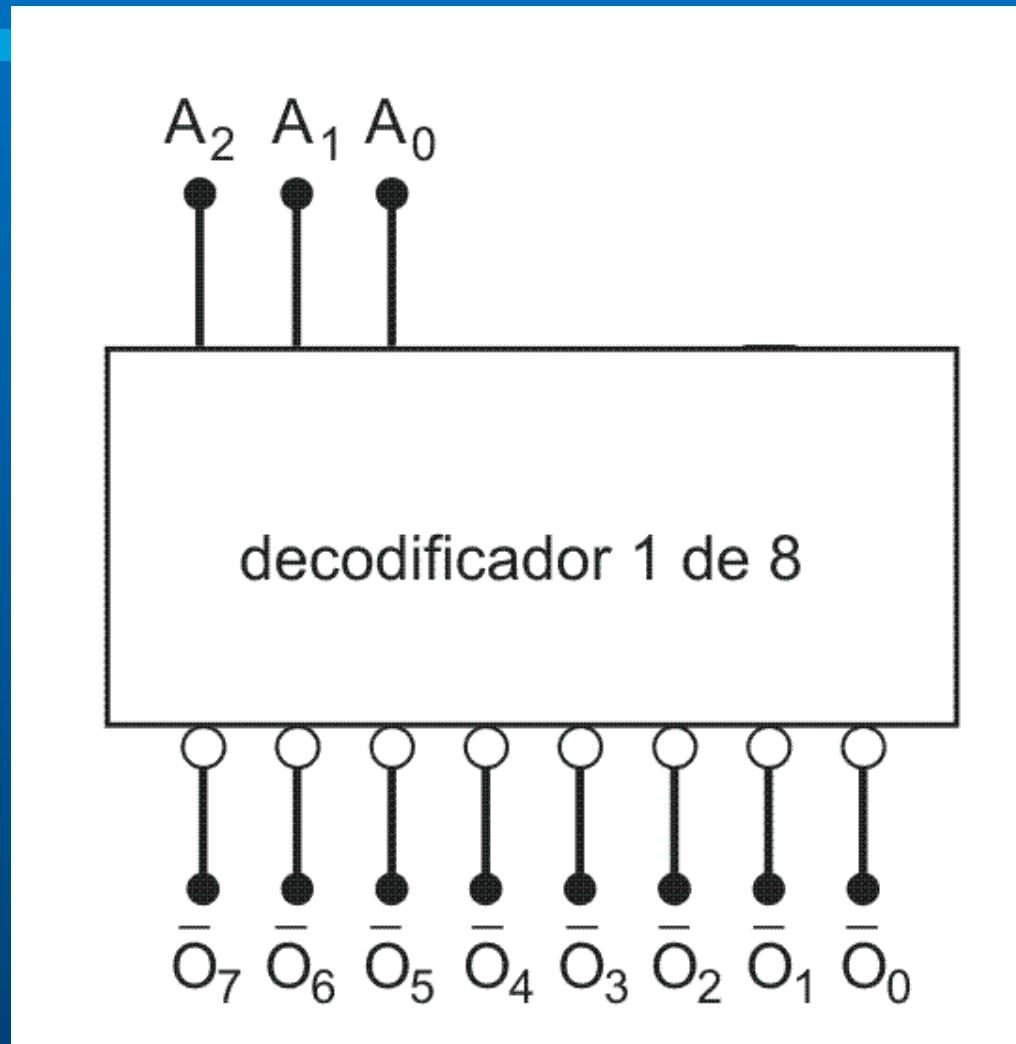
Table [A,B] => X

```
{ 'd'0 => 0;
  'd'1 => 1;
  'd'2 => 1;
  'd'3 => 0; }
```

(c) Decimal set values

# Decodificador 3 x 8

## 9. Exemplos e Aplicações com PLDs



# Decodificador 3 x 8

## 9. Exemplos e Aplicações com PLDs

Table [C,B,A] => [O7,O6,O5,O4,O3,O2,O1,O0]

{	0	=>	'b'00000001
	1	=>	'b'00000010
	2	=>	'b'00000100
	3	=>	'b'00001000
	4	=>	'b'00010000
	5	=>	'b'00100000
	6	=>	'b'01000000
	7	=>	'b'10000000 }

**Note:** Hex values representing inputs

**Note:** Binary values representing outputs

```
/*      Inputs      */
```

```
pin 1 = Azero ;  
pin 2 = Aone ;  
pin 3 = Atwo ;
```

```
/*      Outputs     */
```

```
pin [19..12] = ! [S7..0] ; ;
```

```
/*      SET DEFINITIONS      */
```

```
field inputs = [Atwo, Aone, Azero];  
field outputs = [S7..0]
```

```
/*      Hardware Description      */
```

table	inputs	=>	outputs	
{	[0,0,0]	=>	1;	/* output 0 active */
	[0,0,1]	=>	2;	/* output 1 active */
	[0,1,0]	=>	4;	/* output 2 active */
	[0,1,1]	=>	8;	/* output 3 active */
	[1,0,0]	=>	10;	/* output 4 active */
	[1,0,1]	=>	20;	/* output 5 active */
	[1,1,0]	=>	40;	/* output 6 active */
	[1,1,1]	=>	80;}	/* output 7 active */

## **Decodificador 3 x 8**

**Programado em Assembly (Intel 8051)**

# Decodificador 3X8

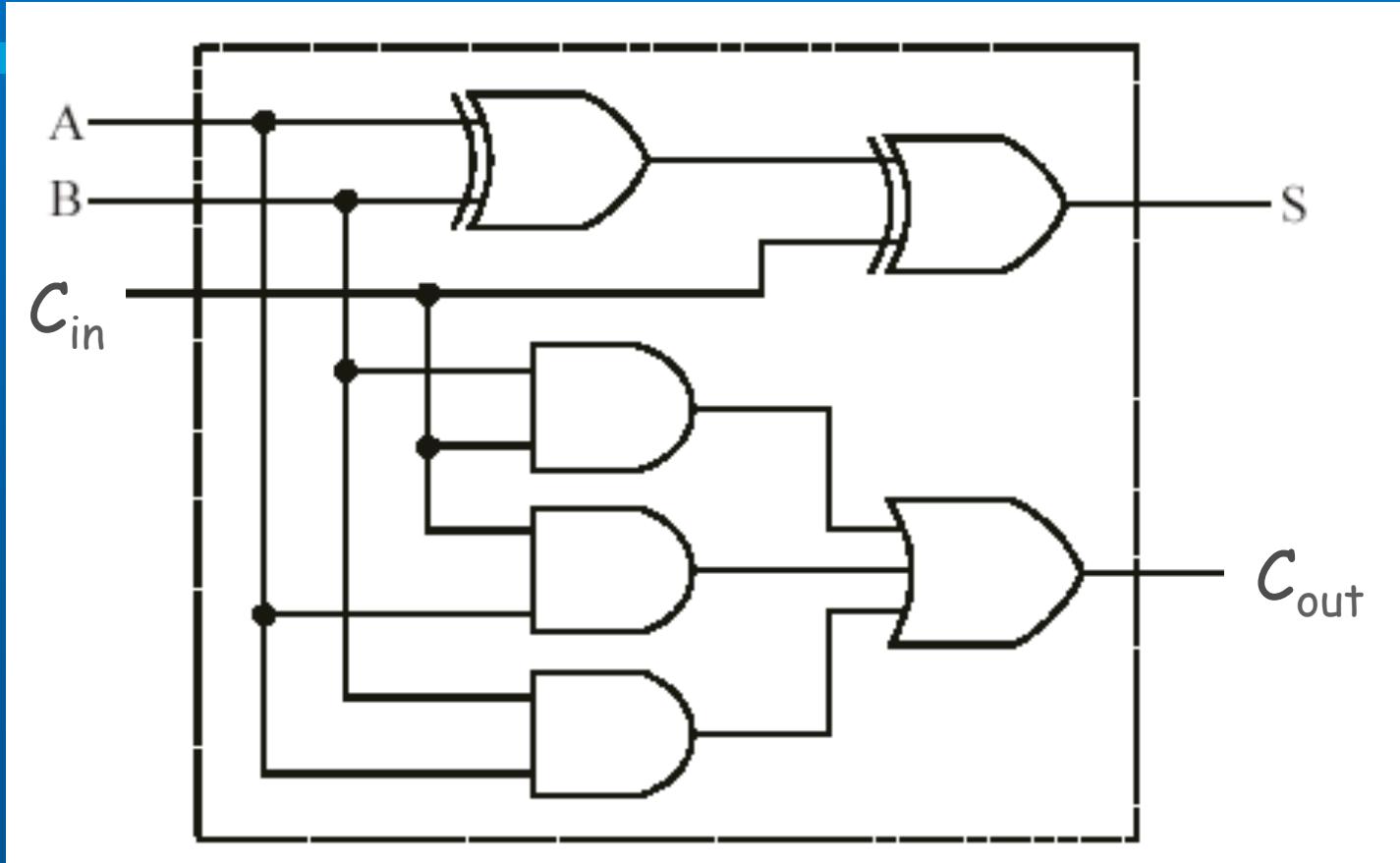
Porta P0 como entrada (0-2) - Porta P1 como saída (0-7)

```
LOOP:  
MOV A,P0  
ANL A, #00000111B  
XRL A, #00000000B  
JZ ZERO  
XRL A, #00000001B  
JZ UM  
XRL A, #00000010B  
JZ DOIS  
XRL A, #00000011B  
JZ TRES  
XRL A, #00000100B  
JZ QUATRO  
XRL A, #00000101B  
JZ CINCO  
XRL A, #00000110B  
JZ SEIS  
XRL A, #00000111B  
JZ SETE  
SJMP LOOP
```

```
ZERO:  
MOV P1, #00000001B  
SJMP LOOP  
UM:  
MOV P1, #00000010B  
SJMP LOOP  
DOIS:  
MOV P1, #00000100B  
SJMP LOOP  
TRES:  
MOV P1, #00001000B  
SJMP LOOP  
QUATRO:  
MOV P1, #00010000B  
SJMP LOOP  
CINCO:  
MOV P1, #00100000B  
SJMP LOOP  
SEIS:  
MOV P1, #01000000B  
SJMP LOOP  
SETE:  
MOV P1, #10000000B  
SJMP LOOP
```

# Somador Completo (1bit)

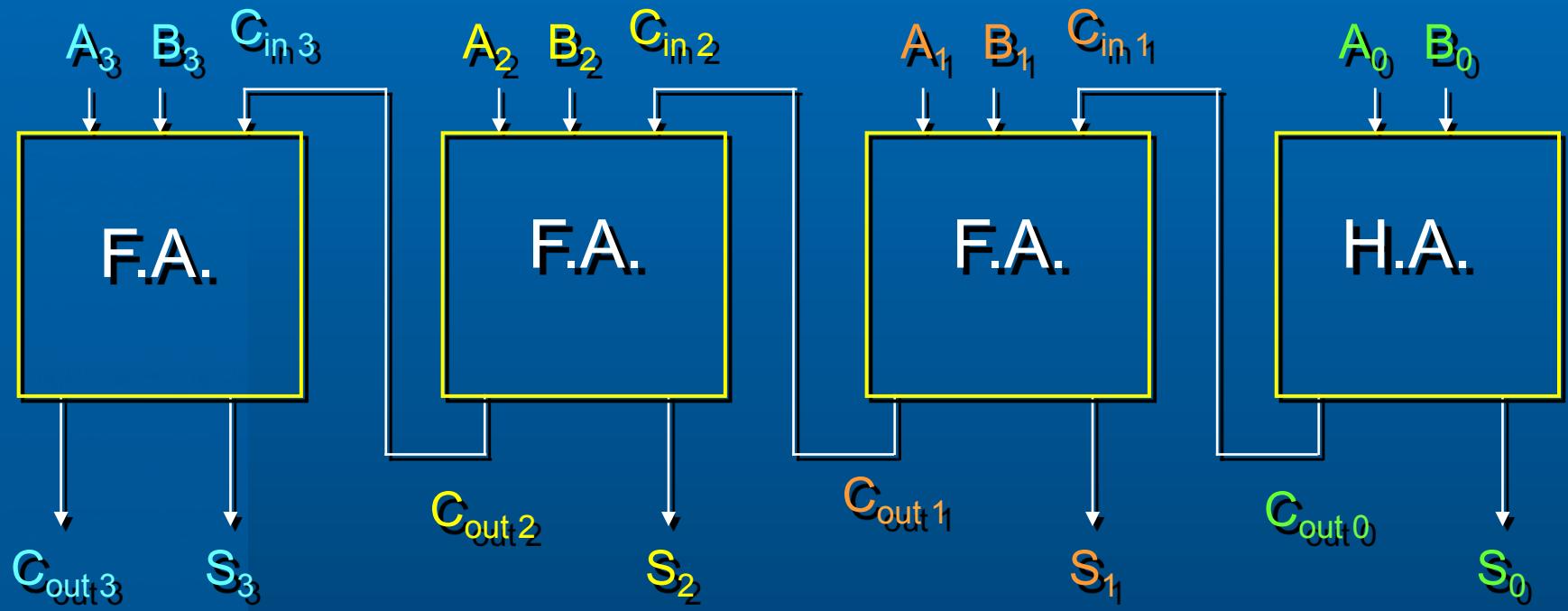
## 9. Exemplos e Aplicações com PLDs



$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

# Somador Completo (4 bits)



## 6. Exemplos e Aplicações

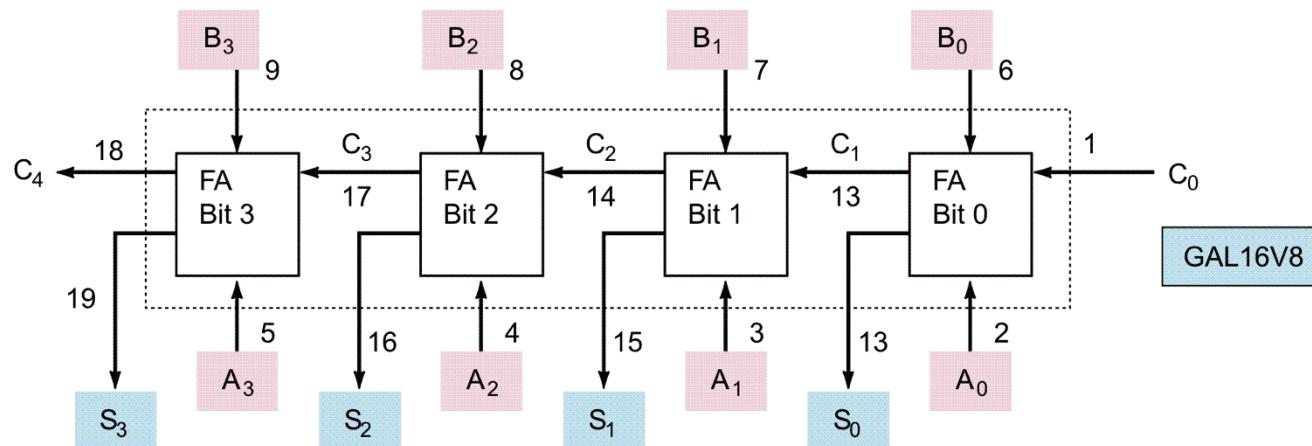
### Somador completo com um PLD

1 <sup>a</sup> PARCELA	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	A
2 <sup>a</sup> PARCELA	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	B
CARRYin	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>	Cin
SOMA	S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	S

Geração da soma  
 $S = A \$ [B \$ Cin]$

1 <sup>a</sup> PARCELA	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	A
2 <sup>a</sup> PARCELA	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	B
CARRYin	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>	Cin
CARRYout	C <sub>4</sub>	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	Cout

Geração dos bits de carry  
 $Cout = A \& B \# A \& Cin \# B \& Cin$



```

/*      4-bit full adder example          */

/*  INPUTS                                */

pin 1 = C0;           /* Carry IN Labeled Carry bit zero*/
pin [2 .. 5] = [A0 .. 3]; /* 4-bit addend A */
pin [6 .. 9] = [B0 .. 3]; /* 4-bit addend B */

/*  OUTPUTS                               */

pin [12, 15, 16, 19] = [S0 .. 3];
pin [13, 14, 17, 18] = [C1 .. 4]; /* Use C4 (pin 18) for carry out of 4-bit
addition */

/*  SET Definitions                      */

field A = [A3 .. 0];           /* 4-bit Augend      */
field B = [B3 .. 0];           /* 4-bit Addend      */
field S = [S3 .. 0];           /* 4-bit Sum         */
field Cin = [C3 .. 0];          /* Carry IN to each of 4 full adders */
field Cout = [C4 .. 1];          /* Carry OUT from each full adder */

/*  Hardware Description                 */

Cout = A&B # A&Cin # B&Cin; /* One equation defines all 4 Carry out
bits */
S = A$ (B$Cin); /* This equation defines the 4-bit set of
the Sum */

```

FIM